Autoregressive Models

Lecture 2

18-789

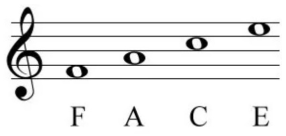
Administrative

- AWS account
- Sign up for student presentations soon!

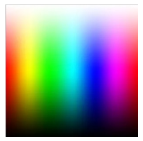
- Today
  - Introduction to autoregressive models
  - Language modeling basics (set up for later lectures)

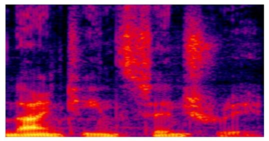
Recap: Probability Distribution

- Generative modeling = modeling the distribution of data
- **Discrete** distributions: the data can only take on certain values
  - Sum of the **probabilities** is 1: $\sum_i P(x_i) = 1$
  - Text, Music Notes



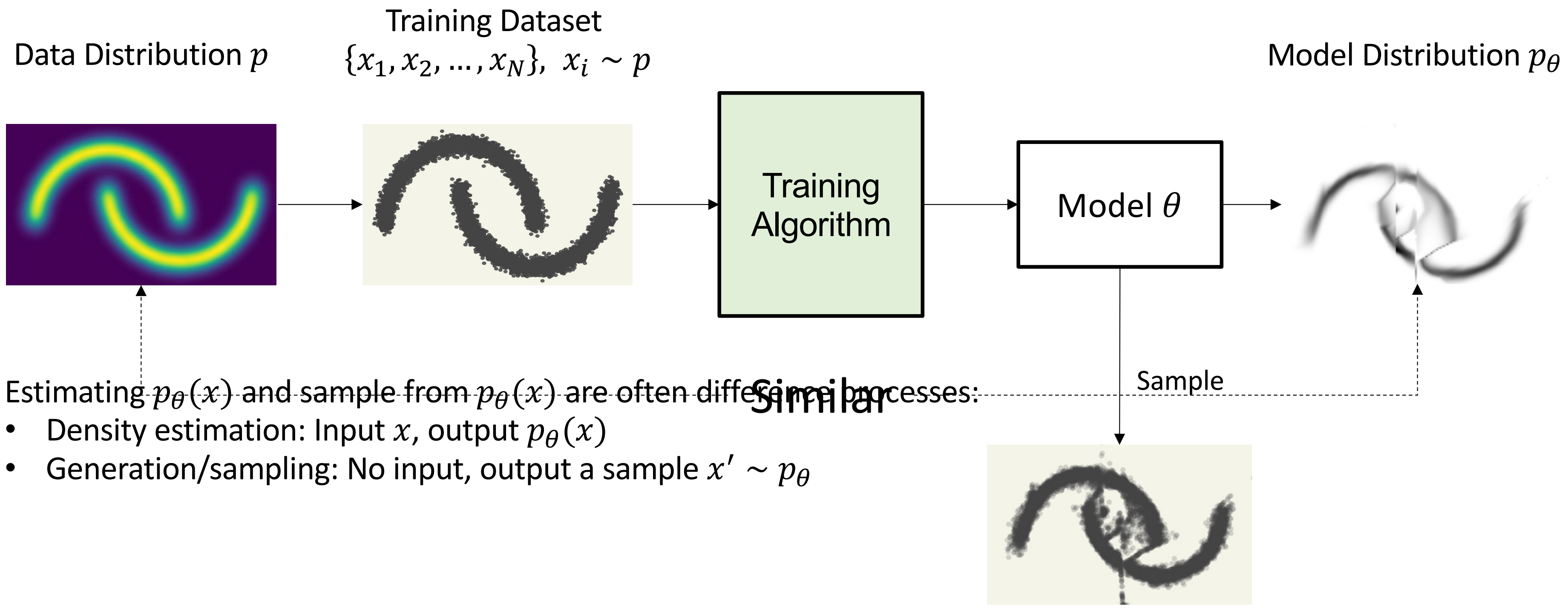
- **Continuous** distributions: the data can take an infinite number values
  - Integral of the probability **density** function is 1: $\int_x p(x)dx = 1$
  - Image, Audio





- Multivariate Distribution: $p(\boldsymbol{x}) = p(x^1, x^2, \ldots, x^D)$

Recap: Generative modeling

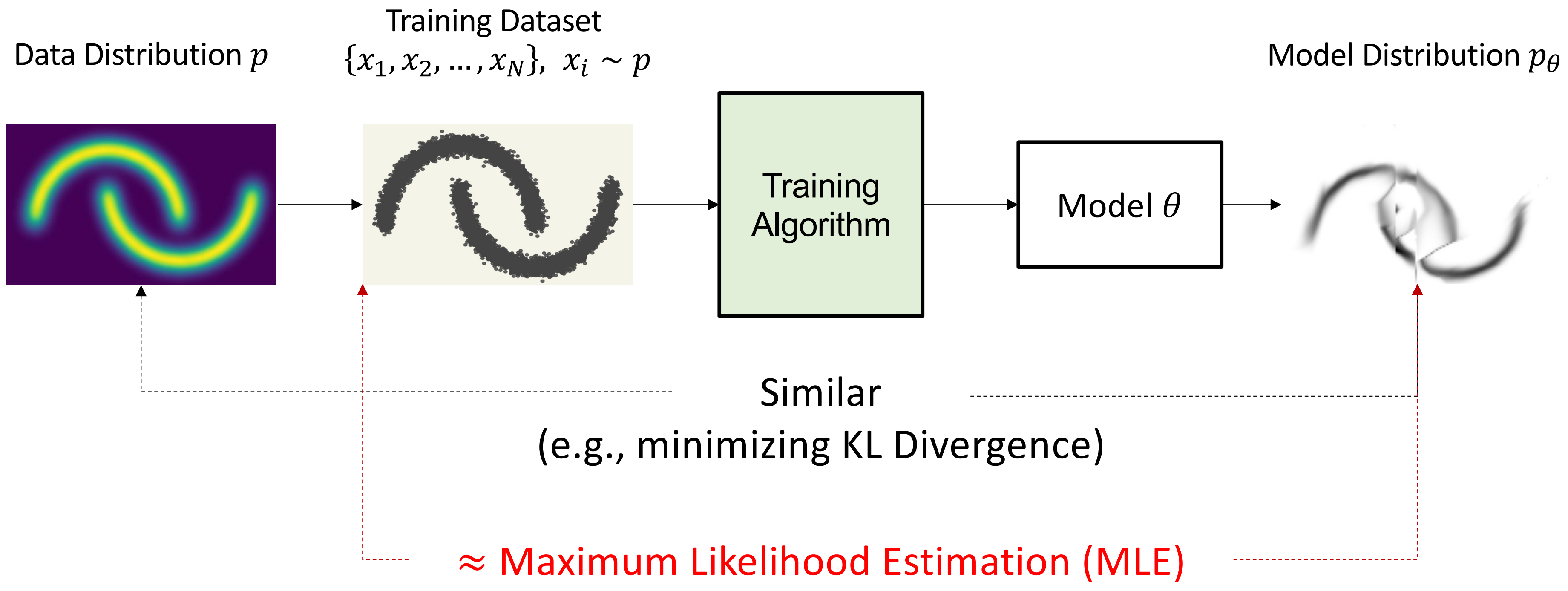


Estimating $p_\theta(x)$ and sample from $p_\theta(x)$ are often difference processes:

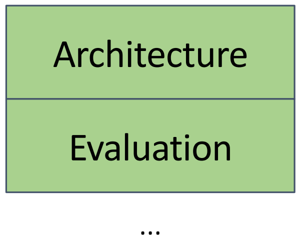
- Density estimation: Input x, output $p_\theta(x)$
- Generation/sampling: No input, output a sample $x' \sim p_\theta$

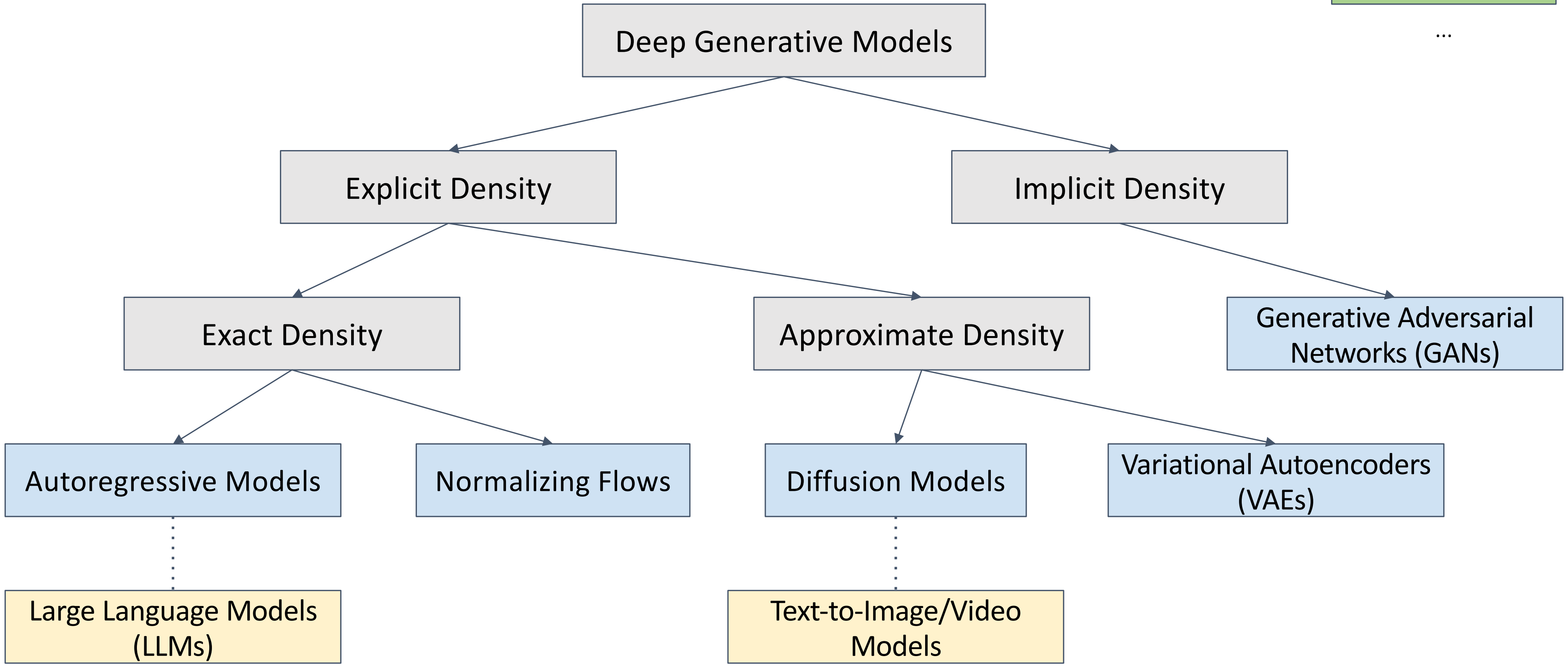
Figures from: https://dmol.pub/dl/flows.html

Recap: Generative modeling


Data Distribution p
Training Dataset
$\{x_1, x_2, \dots, x_N\},\ x_i \sim p$
Training Algorithm
Model θ
Model Distribution p_θ
Similar
(e.g., minimizing KL Divergence)
$\approx$ Maximum Likelihood Estimation (MLE)

Recap: Deep Generative Models

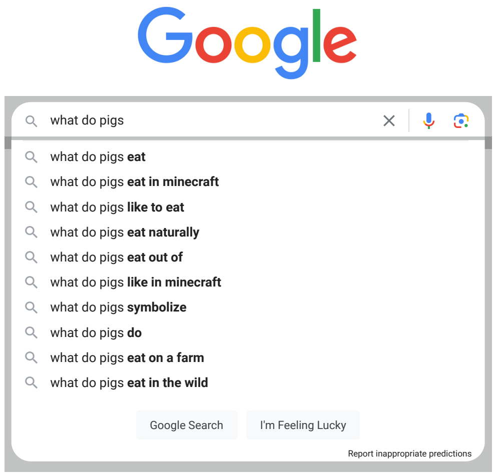




LLMs

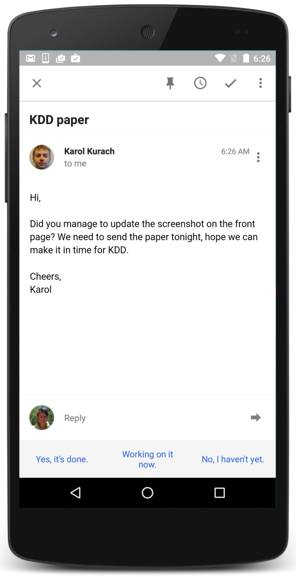
Many NLP tasks

- Natural language generation
- Machine translation
- Speech recognition
- …



what do pigs

what do pigs **eat**
what do pigs **eat in minecraft**
what do pigs **like to eat**
what do pigs **eat naturally**
what do pigs **eat out of**
what do pigs **like in minecraft**
what do pigs **symbolize**
what do pigs **do**
what do pigs **eat on a farm**
what do pigs **eat in the wild**



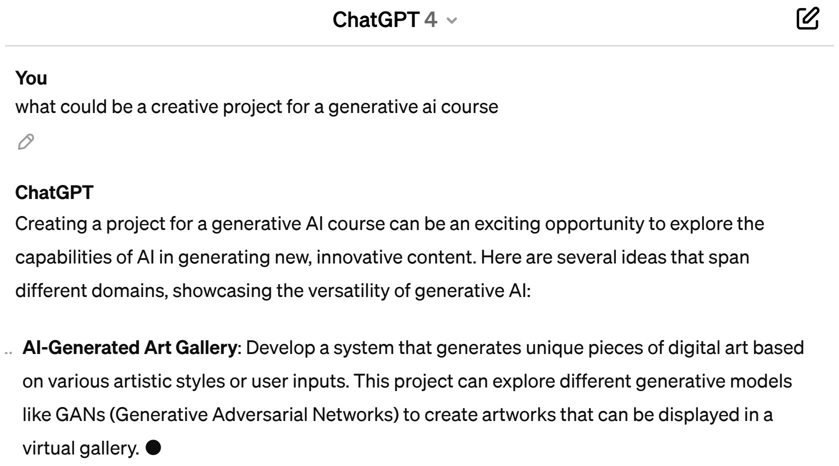
KDD paper

Karol Kurach
to me
6:26 AM

Hi,

Did you manage to update the screenshot on the front page? We need to send the paper tonight, hope we can make it in time for KDD.

Cheers,
Karol



ChatGPT 4

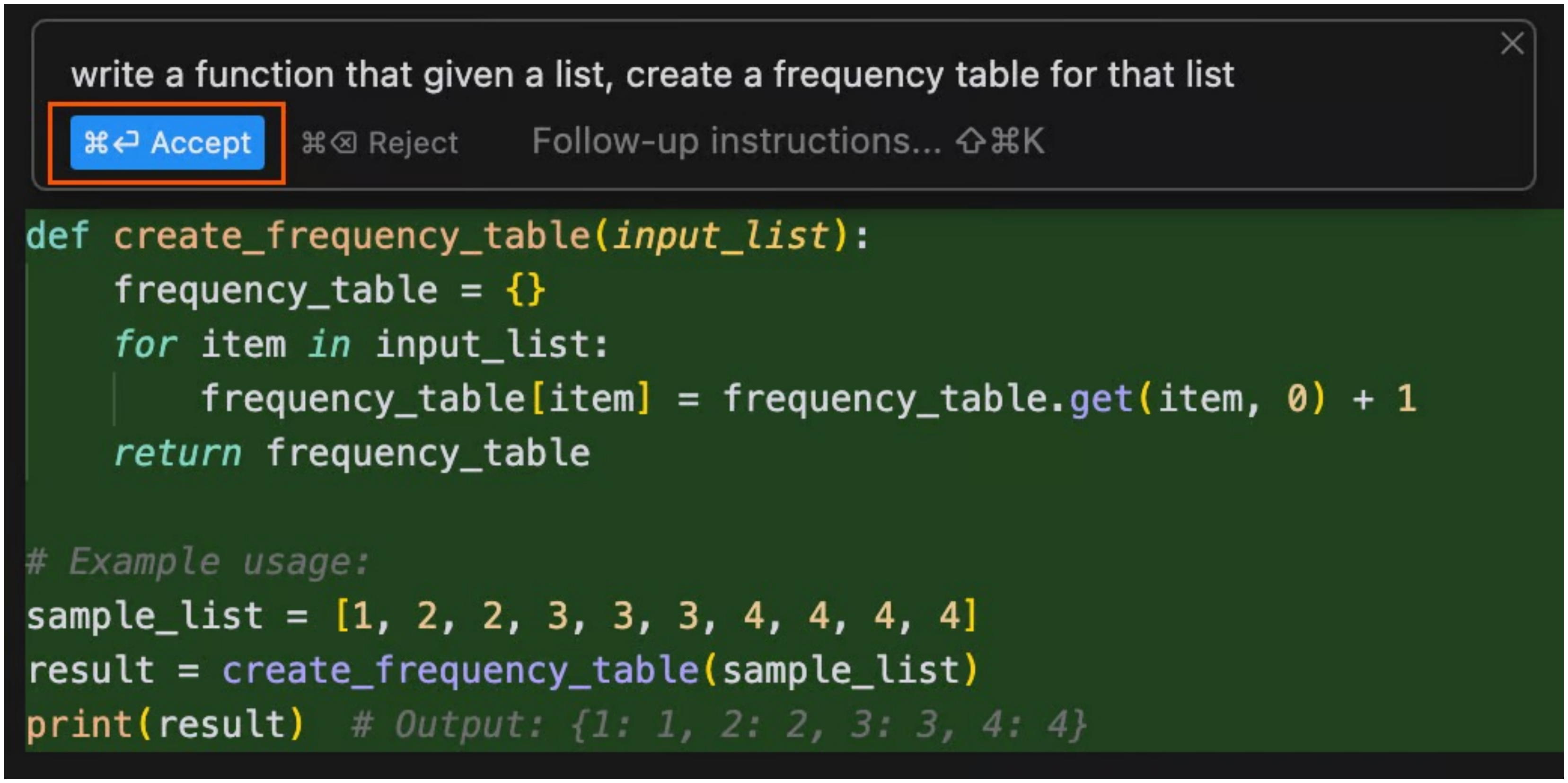
You
what could be a creative project for a generative ai course

ChatGPT
Creating a project for a generative AI course can be an exciting opportunity to explore the capabilities of AI in generating new, innovative content. Here are several ideas that span different domains, showcasing the versatility of generative AI:

AI-Generated Art Gallery: Develop a system that generates unique pieces of digital art based on various artistic styles or user inputs. This project can explore different generative models like GANs (Generative Adversarial Networks) to create artworks that can be displayed in a virtual gallery. ●

Kannan et. Al 2016

Code Generation


write a function that given a list, create a frequency table for that list

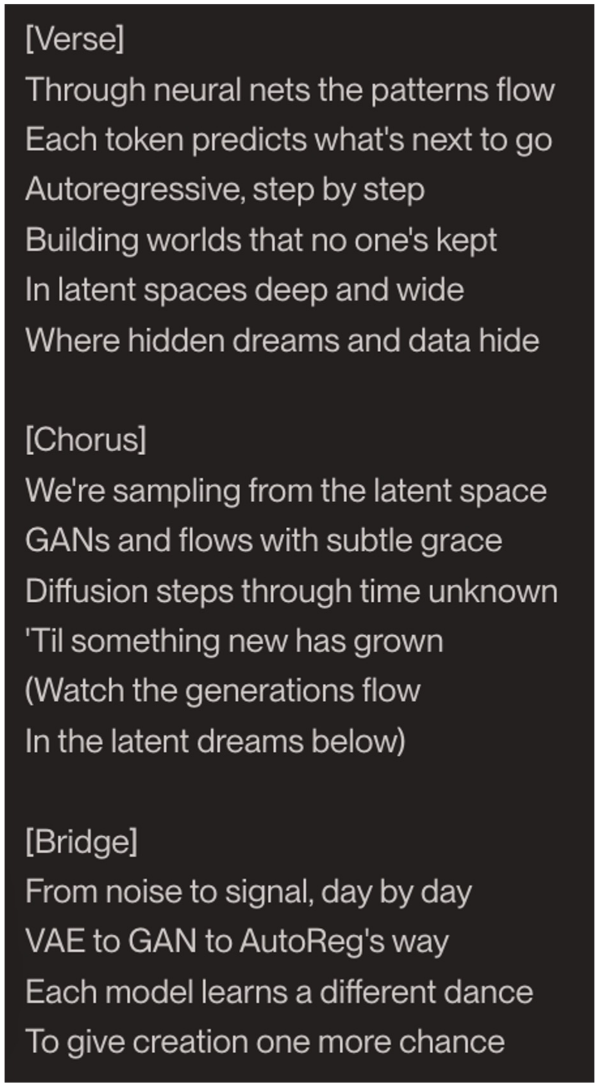
```python
def create_frequency_table(input_list):
    frequency_table = {}
    for item in input_list:
        frequency_table[item] = frequency_table.get(item, 0) + 1
    return frequency_table

# Example usage:
sample_list = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
result = create_frequency_table(sample_list)
print(result)  # Output: {1: 1, 2: 2, 3: 3, 4: 4}
```

Created by Cursor

Music Generation

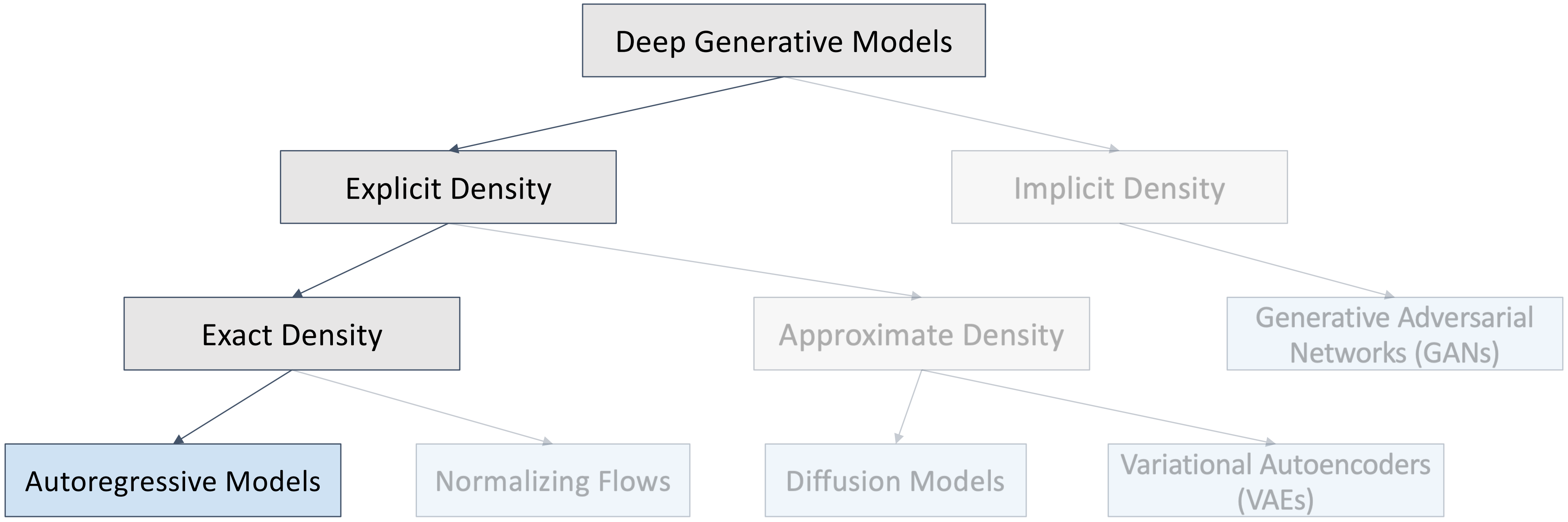
I made a song named “Deep Generative Models” with Suno v4 (lyrics generated by Claude Sonnet 3.5).


[Verse]
Through neural nets the patterns flow
Each token predicts what's next to go
Autoregressive, step by step
Building worlds that no one's kept
In latent spaces deep and wide
Where hidden dreams and data hide

[Chorus]
We're sampling from the latent space
GANs and flows with subtle grace
Diffusion steps through time unknown
'Til something new has grown
(Watch the generations flow
In the latent dreams below)

[Bridge]
From noise to signal, day by day
VAE to GAN to AutoReg's way
Each model learns a different dance
To give creation one more chance

Today: Autoregressive Models



Challenges of Deep Generative Modeling

We want to model the joint distribution of high-dimensional data

- Suppose each pixel is either black or white (Bernoulli distribution) and there are 28x28=784 pixels.

We also want to computational and statistical efficiency

- Efficient training and model representation
- Expressiveness and generalization
- Sampling quality and speed
- Compression rate

Motivation: Estimation by Counting

Recall: the goal is generative modeling is to estimate from

Training Dataset
$\{x_1, x_2, \dots, x_N\}, \; x_i \sim p$

Suppose the samples take on values in a finite set {1,…,100}
The model: a histogram

- Described by $p_1, \dots p_{100}$,
- To train this model: count frequencies
- p_i =

Inference: simple a look up
Sampling:

Model Distribution p_θ

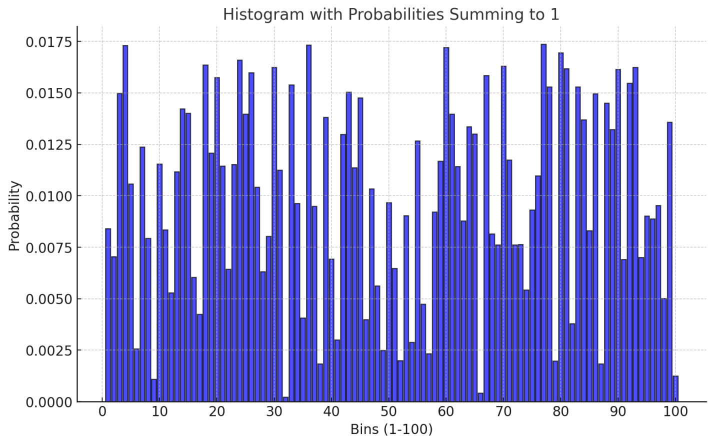


Fig generated by Chat-GPT 4o

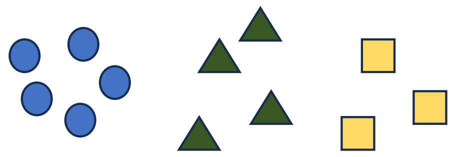
Another Example: Language Model

Language models define probability distributions over the strings in a language.

How likely is a sentence to appear in a language?

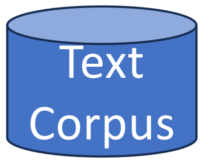
Training Dataset
$\{x_1, x_2, \dots, x_N\}, \; x_i \sim p$

First attempt:



P(What do pigs eat ?) = 0

P(Pigs eat what do ?) = 0



Challenges of Deep Generative Modeling

We want to model the joint distribution of high-dimensional data

- Suppose each pixel is either black or white (Bernoulli distribution) and there are 28x28=784 pixels.
- $2^{784} = 10^{236}$ possible outcomes!
- Training set covers only a Tiny fraction
- Each img influences only one parameter. No generalization

We also want to computational and statistical efficiency

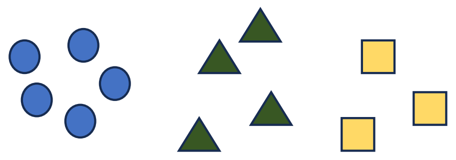
- Efficient training and model representation
- Expressiveness and generalization
- Sampling quality and speed
- Compression rate

Autoregressive Models

Language models define probability distributions over the strings in a language.

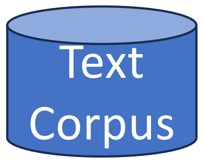
How likely is a sentence to appear in a language?

First attempt:



P(What do pigs eat ?) = 0

P(Pigs eat what do ?) = 0



Sentence probability (chain rule):

P(what do pigs eat ?) = P(? | what do pigs eat) * P(eat | what do pigs) * … * P(what)

P(Y) =

Approach: Autoregressive Model

Left-to-right language models

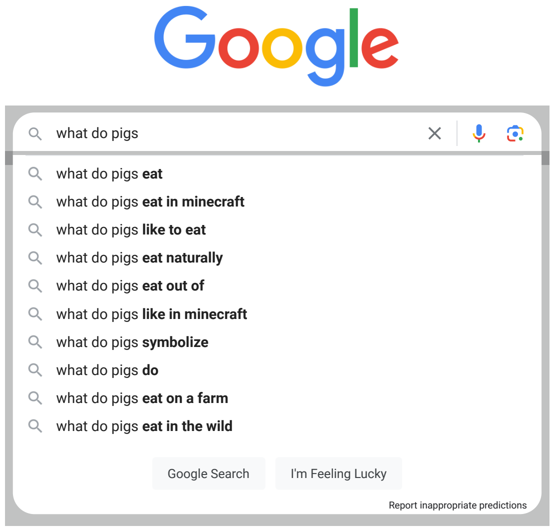
What do pigs __

1. Get probability distribution

Eat 0.5
like 0.3
wear 0.01
…

2. Sample from probability distribution

Eat 0.5
like 0.3
wear 0.01
…



what do pigs

what do pigs **eat**
what do pigs **eat in minecraft**
what do pigs **like to eat**
what do pigs **eat naturally**
what do pigs **eat out of**
what do pigs **like in minecraft**
what do pigs **symbolize**
what do pigs **do**
what do pigs **eat on a farm**
what do pigs **eat in the wild**

N-gram language models

How to compute the conditional probabilities?

An n-gram language model assumes each word depends only on the last n−1 words:

P(what do pigs eat ?) =
P(? | what do pigs eat)
- P(eat | what do pigs)
- P(pigs | what do)
- P(do | what)
- P(what)

$\xrightarrow{\text{3-gram}}$

P(what do pigs eat ?) =
P(? | what do pigs eat)
- P(eat | what do pigs)
- P(pigs | what do)
- P(do | what)
- P(what)

N-gram language models

What if either denominator or numerator is zero?

N(what do pigs) = 0
N(what do pigs do) = 0

What if either denominator or numerator is zero?

- Backoff
- Linear interpolation
- Laplace smoothing

N-gram language models

Left-to-right language models

What do pigs __

1. Get probability distribution

Eat	0.5
like	0.3
wear	0.01
…	

2. Sample from probability distribution

Eat	0.5
like	0.3
wear	0.01
…	

Main Problem?

- Problematic even for unigram
  - Poor generalization
- Inability to use long context

Motivation: Estimation by Counting

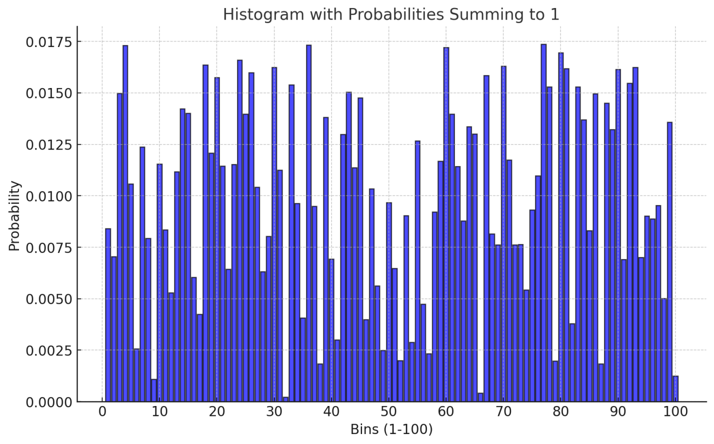
Recall: the goal is generative modeling is to estimate from

Training Dataset
$\{x_1, x_2, \ldots, x_N\},\ x_i \sim p$

Suppose the samples take on values in a finite set {1,…,100}
The model: a histogram

- Described by $p_1, \ldots\, p_{100}$,
- To train this model: count frequencies
- p_{i} =

Model Distribution p_θ



Inference: simple a look up
Sampling:

Fig generated by Chat-GPT 4o

Parameterized Distribution

Recall: the goal is generative modeling is to estimate from

Training Dataset
$\{x_1, x_2, \dots, x_N\},\ x_i \sim p$

Now we introduce function approximation: learn θ so that $p_\theta(x) = p_{data}(x)$

- How do we design function approximators to effectively represent complex joint distribution over x yet remain easy to train
- There will be many designs!

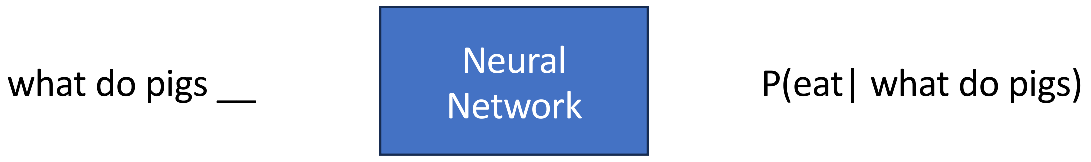
Neural Autoregressive Models

How to compute the conditional probabilities?

- Train a neural network to predict them

1. Process context $\rightarrow$ RNNs, Masking-based Models,

2. Generate a probability distribution for the next token



Tokenizer

Tokenization is the task of taking text (or code or music) and turning it into a sequence of discrete items, called tokens.

- Word-based
- Character-based
- Subword-based

Tokenizer

Tokenization is the task of taking text (or code or music) and turning it into a sequence of discrete items, called tokens.

Word-based

[What, do, pigs, eat, ?]

- Too large vocabulary (260000+)
- Rare word collapse

Tokenizer

Tokenization is the task of taking text (or code or music) and turning it into a sequence of discrete items, called tokens.

Character-based

[W, h, a, t, d, o, p, l, g, s, e, a, t, ?]

- 256
- Harder to learn
- Long context

Tokenizer

Tokenization is the task of taking text (or code or music) and turning it into a sequence of discrete items, called tokens.

Subword-based [What, do, dog, s, eat, ?]

- Best of both world

WordPiece, Unigram, BPE (Byte-Pair Encoding)

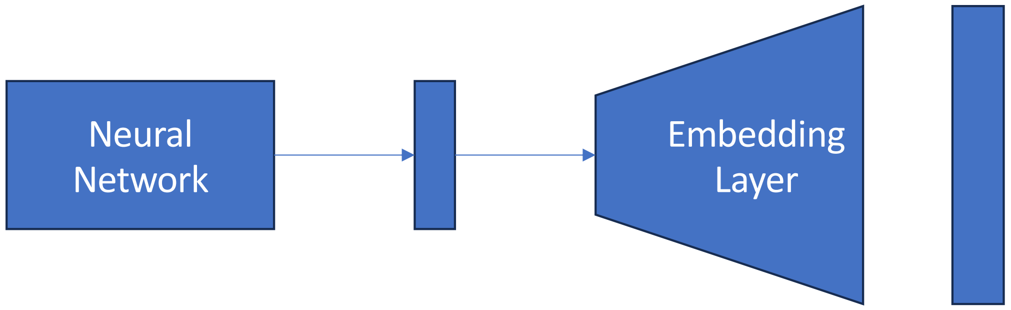
Bert ALBERT GPT

e.g., BPE ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

Training Pipeline

Forward pass

- Feed word embedding for previous (context) words into a network
- Get vector representation of context from the network
- From this vector representation, predict a probability distribution for the next token



$P(y_T \mid y_{<T}) =$

Training Pipeline

Cross-entropy loss:

Loss (p*, p) =

KL divergence

Decoding / Inference

How can we sample from a distribution?

What do pigs __

Eat	0.5
like	0.3
wear	0.01
…	

1. Argmax?
2. Random sample?

Problems?

Decoding / Inference

How can we sample from a distribution?

What do pigs __

Eat	0.5
like	0.3
wear	0.01
…	

1. Argmax?
2. Random sample?
3. Sample with temperature!
4. Top-k sampling!
5. Top-p sampling!
6. Beam search!

Beam search

Assumption: the best possible sequence to generate is the one with highest overall sequence likelihood (according to the model).

But it’s intractable …

Beam search is a greedy algorithm which approximates finding the overall most likely sequence to generate!

Problems?

- Non-creative …

How Can We Choose Models?

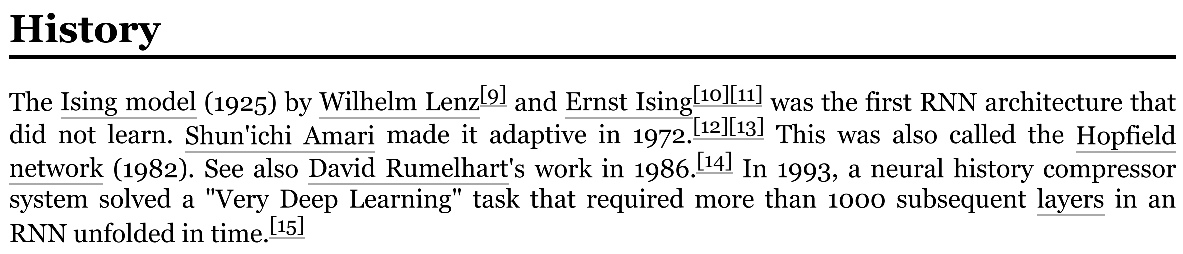
Two variables: y_1, y_2
Model: $p(y_1, y_2) = p(y_1)p(y_2|x_1)$

Does it extend to high dimensions?

- For d-dim data, O(d) parameters
  - But what about d is really large?
- Limited generalization
  - No info shared among different conditions

Fig generated by Chat-GPT 4o

Recurrent neural networks



History

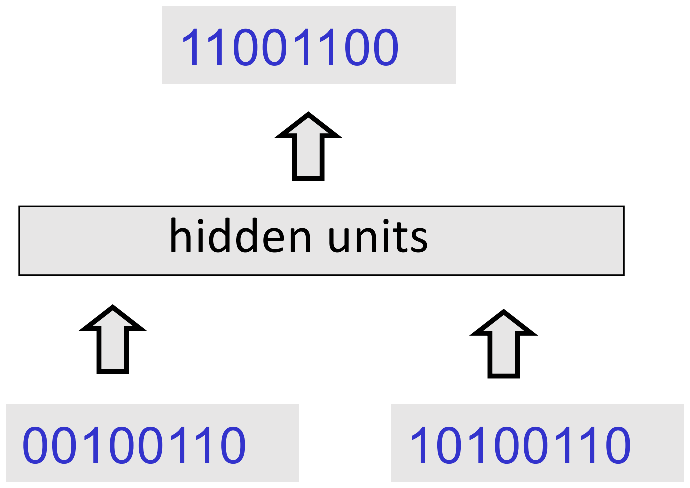
The Ising model (1925) by Wilhelm Lenz[9] and Ernst Ising[10][11] was the first RNN architecture that did not learn. Shun'ichi Amari made it adaptive in 1972.[12][13] This was also called the Hopfield network (1982). See also David Rumelhart's work in 1986.[14] In 1993, a neural history compressor system solved a "Very Deep Learning" task that required more than 1000 subsequent layers in an RNN unfolded in time.[15]

Why RNNs? A toy example:

Problem with memoryless (FFN)?

- Maximum number of digits
- Processing at the beginning of a long

Number does not generalize to the end

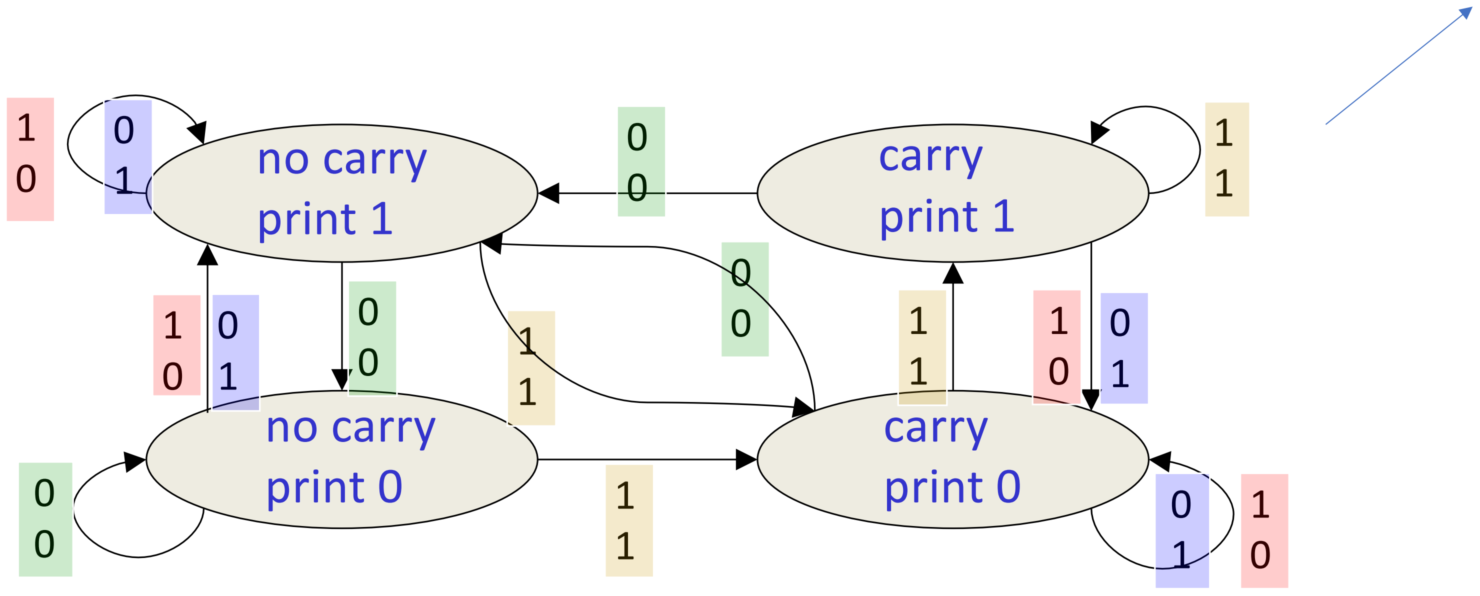


Hinton et al.

Recurrent neural networks

The algorithm for binary addition:

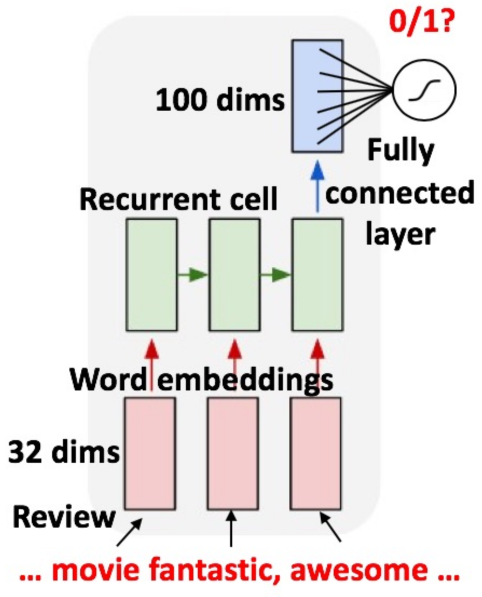
Finite state automaton

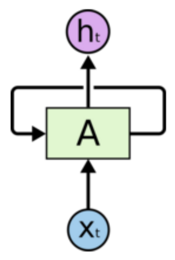


Hinton et al.

Recurrent neural networks

- Recurrent Neural Networks are networks with loops, allowing information to persist.





Output is to predict a vector h_t, where $output\ y_t = \varphi(h_t)$ at some time steps (t)

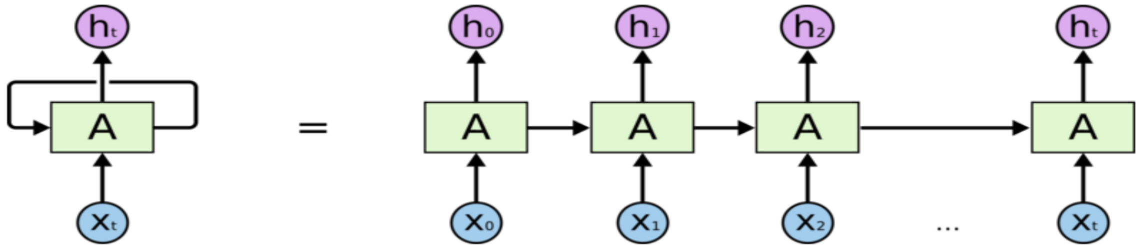
Recurrent Neural Networks have loops.

In the above diagram, a chunk of neural network, ***A*** = ***f****W*, looks at some input ***x****t* and outputs a value ***h****t*. A loop allows information to be passed from one step of the network to the next.

$$\overset{\text{new state}}{\boxed{h_t}} = \underset{\text{function with parameter } W}{\boxed{f_W}}(\overset{\text{old state}}{\boxed{h_{t-1}}}, \underset{\text{Input vector at some time step}}{\boxed{x_t}})$$

Recurrent neural networks

- Unrolling RNN

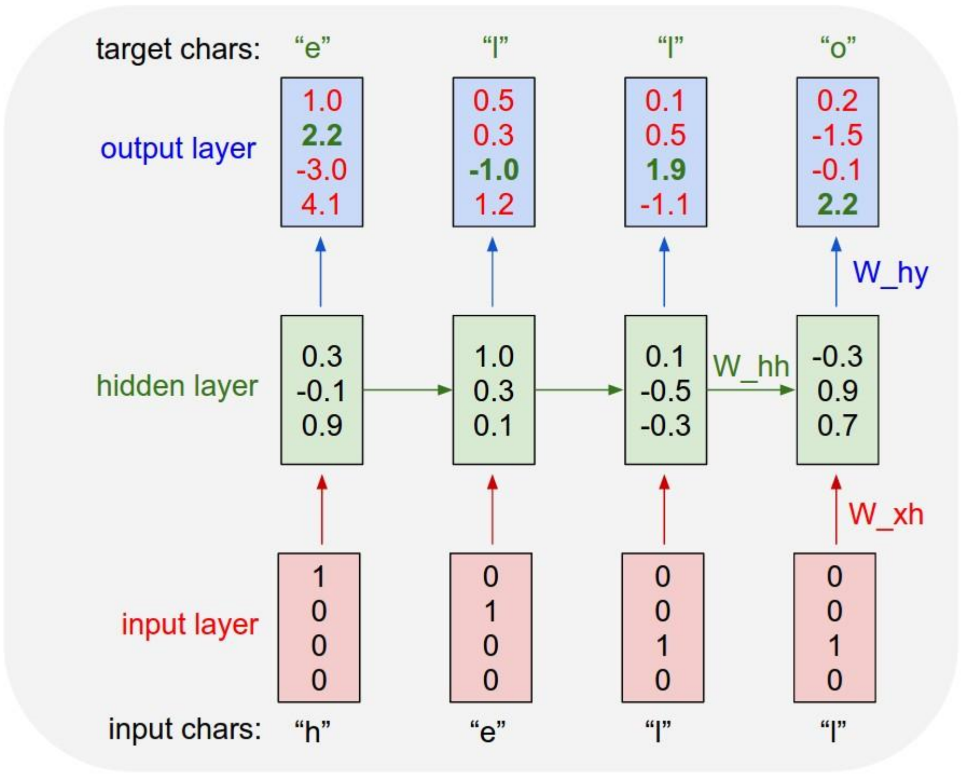


An unrolled recurrent neural network.

A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. The diagram above shows what happens if we **unroll the loop**.

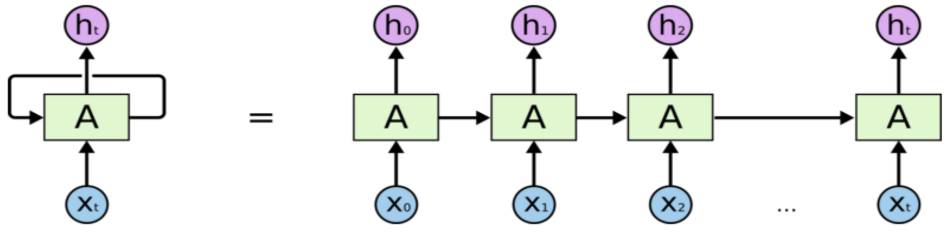
Recurrent neural networks

- Example



Recurrent neural networks

For binary addition task



An unrolled recurrent neural network.

0 0 1 1 0 1 0 0

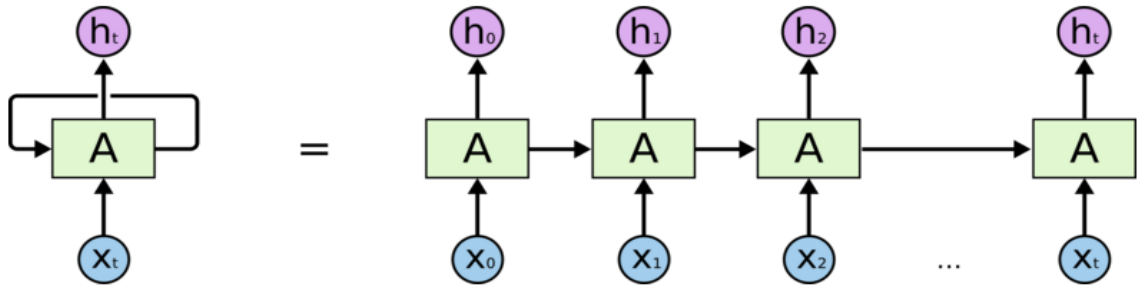
0 1 0 0 1 1 0 1

1 0 0 0 0 0 0 1

time

- The weight from input layer to hidden which the shape is (2,16)
- The weight from hidden layer to output, which the shape is (16,1)
- The weight from hidden layer to hidden layer, which the shape is (16,16)

Back propagation through time



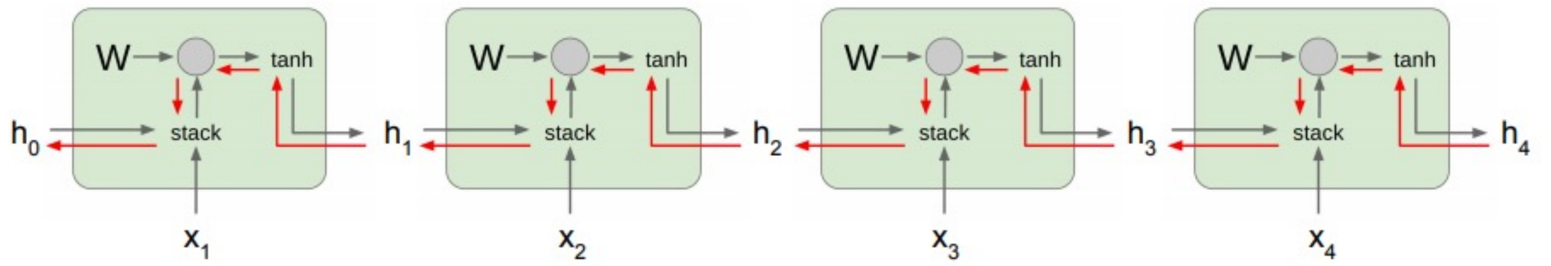
An unrolled recurrent neural network.

Recurrent Neural Networks

The recurrent structure of RNNs enables the following characteristics:

- Specialized for processing a sequence of values $x^{(1)}, \ldots, x^{(\tau)}$
  - Each value $x^{(i)}$ is processed with the **same network *A* that preserves past information**
- Can scale to much **longer sequences** than would be practical for networks without a recurrent structure
  - Reusing network ***A*** reduces the required amount of parameters in the network
- Can process **variable-length sequences**
  - The network complexity does not vary when the input length change
- However, vanilla RNNs suffer from the training difficulty due to **exploding and vanishing gradients**.

Exploding and Vanishing Gradients



In vanilla RNNs, computing this gradient involves many factors of W_{hh} (and repeated $\tanh$)*. If we decompose the singular values of the gradient multiplication matrix,

- Largest singular value > 1 → **Exploding gradients**
    - Slight error in the late time steps causes drastic updates in the early time steps → Unstable learning
- Largest singular value < 1 → **Vanishing gradients**
    - Gradients passed to the early time steps is close to 0. → Uninformed correction

* Refer to Bengio et al. (1994) or Goodfellow et al. (2016) for a complete derivation

5-minute quiz

Password: pig